DoS Attack Impact Assessment on Software Defined Networks

Abimbola Sangodoyin[1], Tshiamo Sigwele[1], Prashant Pillai[2], Yim Fun Hu[1], Irfan Awan[1] and Jules Disso[3]
[1]Faculty of Engineering and Informatics, University of Bradford, Bradford, BD7 1DP, UK
{a.o.sangodoyin, t.sigwele, i.awan, y.f.hu}@bradford.ac.uk
[2]Faculty of Technology, Design and Environment, Oxford Brookes University, Oxford, UK
ppillai@brookes.ac.uk
[3]Nettitude Limited, United Kingdom, Warwickshire, CV31 3RZ
jpagnadisso@nettitude.com

***Abstract*—Software Defined Networking (SDN) is an evolving network paradigm which promises greater interoperability, more innovation, flexible and effective solutions. Although SDN on the surface provides a simple framework for network programmability and monitoring, few has been said about security measures to make it resilient to hitherto security flaws in traditional network and the new threats the architecture is ushering in. One of the security weaknesses the architecture is ushering in due to separation of control and data plane is Denial of Service (DoS) attack. The main goal of this attack is to make network resources unavailable to legitimate users or introduce large delays. In this paper, the effect of DoS attack on SDN is presented using Mininet, OpenDaylight (ODL) controller and network performance testing tools such as iperf and ping. Internet Control Message Protocol (ICMP) flood attack is performed on a Transmission Control Protocol (TCP) server and a User Datagram Protocol (UDP) server which are both connected to OpenFlow switches. The simulation results reveal a drop in network throughput from 233Mbps to 87.4Mbps and the introduction of large jitter between 0.003ms and 0.789ms during DoS attack.**

***Index Terms*—Software Defined Networks, DoS, Network Security.**

I. Introduction

Computer networks have become part of our everyday lives from government to commercial enterprises to individuals. [1]. These networks are built from large number of devices such as routers, switches and middle boxes with complex protocols running on them. Network administrators are saddled with the responsibility of configuring these devices that are vendor specific and configuration policies are implemented on each devices. As a result, network management and dynamic response to events and applications is arduous and prone to error.

In addition to configuration complexity, operators have little options or mechanisms to respond to difficulties and enforce the required policies in dynamic environments [2]. Similarly, in the face of growing traffic and demand for more data rate from consumers, the service providers must keep up with the pace by investing in bigger and faster links and edge routers, even though revenues are growing quite slowly [3].

In view of the challenges network operators are facing, the need for a programmable network which is cost effective and robust enough to meet the demand of users is imperative. Thus, the emergence of Software Defined Networking (SDN). SDN has created great hope to overcome age-old problems in networking while simultaneously enabling the introduction of complex, secure and reliable network policies for next generation networks [4]. The revolutionary concept of SDN has brought change to existing networks by separating the forwarding functionalities of existing devices, known as data plane from control element, known as control plane [5].

The future of SDN lies solely in its acceptance and deployment. Technology and its deployment take years before it can be available to end users due to standardisation process and Request for Comments (RFCs). Speculatins however remain as to whether same should be expected for SDN or not. According to [1] a proposal for open and programmable network is presented. The author further emphasize the need for researchers to run experiment on campus network using an OpenFlow switch. In line with this, ETHANE, a new network architecture for enterprise was suggested [6]. In their proposed architecture, ETHANE switch doesnt need to learn addresses, support Virtual Local Area Networks (VLANs) or check for source-address spoofing and it has been deployed in a campus environment. The proposal was taken to another level when a major player in the networking industry, Google, deployed B4 using OpenFlow switches in their Wide Area Networks (WAN) data centre [7]. Also, with the advent of a Linux foundation collaborative project, OpenDaylight (ODL) [8] platform and VMware NSX virtualisation platform [13], global acceptance and deployment is not far from reach.

In spite of the programmability, flexibility, universal connectivity and decentralised control, which were critical to SDN success, they are at odds with making it more secure. The SDN platform can bring with it several security breaches which include an increased potential for Denial-of-Service (DoS) attacks due to controller centralisation and flow table limitations in network devices [9]. Furthermore, abstraction of flows and underlying hardware resources make it easier for harvesting of intelligence which can be used effortlessly for further exploitation and reprogramming entire network by malicious user [4].

In this paper, the impact of DoS attack on SDN is presented. The simulation has been perfomed using mininet and Open-

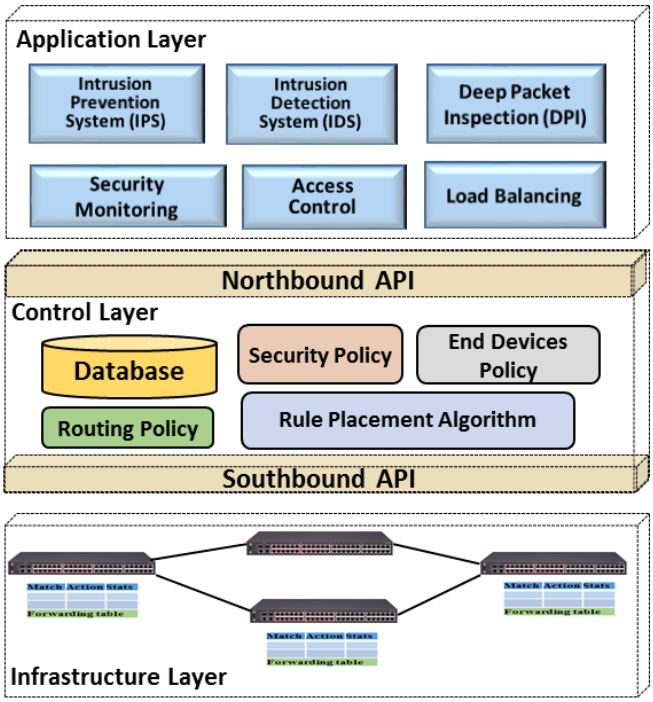

Fig. 1. SDN architecture illustrating the data, control and application layers.

Daylight controller tools and the simulation result shows that DoS flooding attack on SDN network can degrade network performance by decreasing network throughput and introduce large jitter. This paper is structured as follows: Section II presents related works on the SDN architecture, vulnerabilities in the SDN architecture and DoS attacks on SDN. The experimental method and tools are presented in Section III. Then Section IV shows the experimental set-up. The results and analysis are presented in Section V. Finally, the conclusion and future work are presented in Section VI.

II. Related Work

A. SDN Architecture

SDN architecture encompasses the complete network platform. It is a modular approach that defines chain of command and interoperability within network. Unlike traditional network, the intelligence of data plane devices is removed to a logically centralised control system [10]. Fig. 1 presents the SDN architecture showing the data/infrastructure, control and application layer. In an SDN architecture, there are two main elements: the controllers and the forwarding devices. A forwarding device is a hardware or software element specialised in packet forwarding and based on a pipeline of flow tables where each entry of a flow table has: a matching rule, action to be executed on matching packets and counters that keep statistics of matching packets [4]. The controller serves as the brain of the network and it deals with management of network state. Below is a description of various layers:

Infrastructure layer: This layer is also known as data plane. It consists of simple forwarding elements without embedded control or software to take autonomous decisions. It is accessible through the southbound interface and allows packet switching and forwarding.

Control layer: This layer consists of SDN controllers providing a consolidated control functionalities through Application Programming Interfaces (APIs). The crucial value of the controller is to provide abstractions, essential services, and common APIs to developers. Three communication interfaces allows the controller to interact: northbound, southbound and the east/westbound interfaces.

i) Southbound Interfaces: Southbound interface allows the controller and forwarding elements to interact in the infrastructure layer, thus being the crucial instrument for clearly separating the control and data plane functionality.
ii) Northbound Interfaces: This interface is the connecting bridge between application layer and control layer. It enables the programmability of the controllers by exposing the data models and other functionalities within the controllers for use by applications at the application layer. The northbound interface is mostly a software ecosystem, hence, a common northbound interface is still an open issue.
iii) East/Westbound Interfaces: This interface is a special communication interface envisioned for distributed controllers to synchronise state for high availability. Its function include import/export data between controllers and monitoring/notification capabilities to check if a controller is up or notify a takeover on a set of forwarding elements.

Application Layer: The application layer consists of end-user business applications and network services. Example of application that runs here is network virtualisation. Network policy is also defined here.

B. Vulnerabilities in SDN Architecture

A number of security analyses has been carried on the vulnerabilities in SDN. Adnan et al. in [5] identified the state of art in SDN security solutions with respect to each layer of SDN architecture. The work focuses on possible security attacks in SDN which could be executed. However, no solution to identified threats is presented. A comprehensive survey of security in SDN is presented in [11][12], the authors identified vulnerabilities introduced by separation of control and data plane. Sandra et al. in [11] presents an overview of SDN security and itemise research work coupled with solution to security issues in SDN. In [12] classification is done using the STRIDE approach and possible SDN security controls is proposed. The concept of offering SDN security as a service is presented in [13].

Kreutz et al. in [2] presents a high level security analysis. Seven main potential threat vectors are presented. Three of the seven identified threat vectors are specific to SDN and relates to the three planes present in SDN architecture. The analysis does not present SDN as a less secure network but triggers the need for innovative ways of responding to the new threats arising from network programmability. The authors state the

consequences of these threats in SDN and solutions to the seven threat vectors was proposed.

In [14], a feasibility study on attacking SDN network is carried out by fingerprinting to know whether the network uses SDN/OpenFlow switches. The SDN network is then subjected to a specifically crafted flow requests from the data plane to the control plane to exhaust the network resources. Another security vulnerabilities was analysed in ProtoGENI [15]. The authors explored three potential security issues as follows:

i) **Resource connection:** Once a malicious user obtains access to one experiment node, attacks can easily be launched by utilising the huge ProtoGENI computing resources as a launchpad to harm existing internet users.
ii) **Wireless Nodes Distribution:** Network sniffing or spoofing can be done here to identify desired node for launching attacks.
iii) **Virtualization Technology:** In ProtoGENI virtualisation, ProtoGENI resources are shared among as many user as possible. Any bug or compromise from a single device will expose other users in sharing resources to attacks.

The Authors discovered the possibility of using ProtoGENI resources to launch flooding attack to the wider internet. Also, the possibility of compromising confidentiality and availability of other ProtoGENI users is high.

C. DoS Attack on SDN

DoS and Distributed DoS (DDoS) attack remains one of the severe network security problems in both traditional network and SDN. Due to separation of control and data plane, an attacker could saturate the controller with malformed packets requiring a flow rule decision. On the other hand, the flow table of the infrastructure device can be overwhelmed with malicious packets. To address bottlenecks of potential saturation attack, AVANT-GUARD [16] introduce connection migration to reduce amount of data-to-control-plane interactions. The method enables the data plane to shield the control plane from saturation attacks. However, the data plane itself is subject to attack. Similarly, a backup strategy which offers resilience against failures in a centralised controlled network is presented in [17]. This approach is an attempt to solve single point of failure bottleneck and it provides seamless transition between primary controller to a back-up controller. However, this solution is limited to centralised implementation and it also raises concern in terms of trust between the east-west interface communications. In addition, Braga et al. [18] proposed lightweight, a new method for detecting DDoS. The proposed method boasts of high rate of true positives and low rate of false alarm using Self Organising Maps (SOM) for flow analysis. The lightweight method consider median values in training the SOM. However, the method reports false negatives when the attack rate parameter is set to a low value.

The controller has been compared to an operating system capable of managing applications through programmatic interface [19]. Similarly, ETHANE was built to provide network-wide fine-grain policy using a centralised declaration and enforcing it [6]. While the concept of a centralised controller allow the simplification of policy enforcement and management tasks for network managers, it creates quite a number of bottle necks. In [9], analysis of SDN implementation key challenges has been carried out. The authors opined deployment of SDN technology will contribute to the vision of future communications if outstanding challenges were resolved. In [20] the possibility of DoS attacks and poor rule design that can lead to saturating volumes of controller queries is discussed. The author highlights the OpenFlow vulnerabilities in terms of lack of adoption of Transport Layer Security (TLS) for controller-switch communication. Although a number of vulnerabilities is proposed, none is proven in the work.

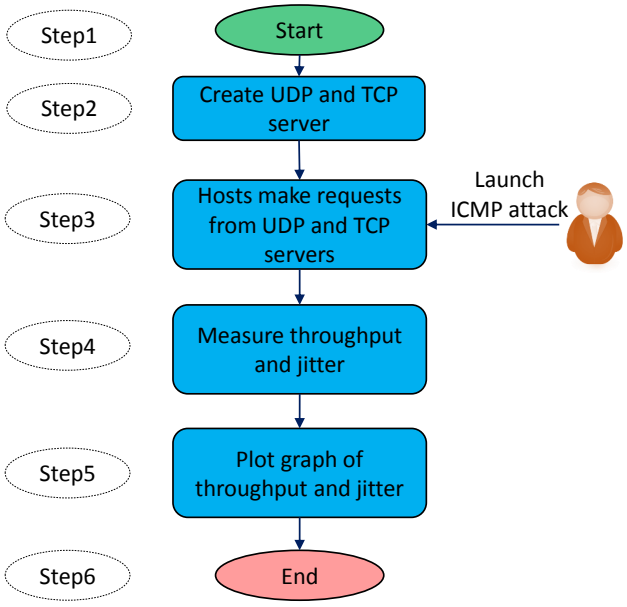

Fig. 2. Methodology flowchart.

III. Experimental Method and Tools

In this experiments , Mininet is used [21]. Mininet is an open source network emulator devoted entirely to OpenFlow architecture and SDN. For the controller, ODL controller is used [22]. ODL integrates open source, open standards and open APIs to deliver SDN platform to make networks more programmable and adaptive.DoS attacks usually engage numerous compromised hosts and a rich topology to launch a successful attack on its victim. While our scenario is much simpler than what is obtainable in real world attacks, we deliberately chose such a low-complexity set-up to expose and analyse the impact of DoS attack on SDN. Common testing tools such as, $ping$ and $iperf$ are used to generate traffic between host and servers. Fig. 2 shows the methodology flowchart with each step explained below:

Step 1: Start Mininet and ODL controller

```
Sudo mn --custom scenario.py --topo
--controller = remote,ip = x.x.x.x
```

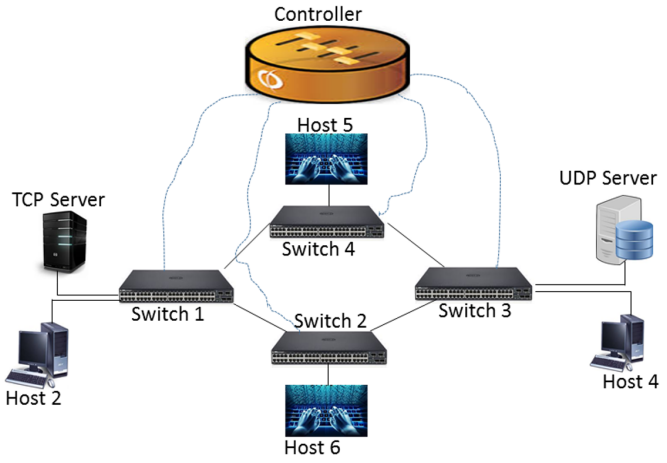

Fig. 3. Expermental setup.

Where x.x.x.x represents the ip address of the remote controller.
Check connectivity using

```
$mininet > net
```

Step 2: Create UDP and TCP server

```
UDP: iperf -s -u -p 5566 -i1
TCP: iperf -s  -p 5566 -i1
```

The TCP server is made to listen on port 5566 with a default window size of 85.3Kilobytes. Similarly, UDP server is made to listen on port 5566 with a default UDP buffer size of 208Kilobytes while receiving 1470 bytes datagrams and the result is monitored every 1 seconds.

Step 3: Hosts make requests from TCP server and UDP server

```
TCP: -iperf -c x.x.x.x -p5566 -t100
UDP: -iperf -c x.x.x.x -u -t100 -p5566
```

Step 4 and Step 5: Results were extracted using AWK file and results plotted using MATLAB. Then, malicious hosts 5 and 6 launched flooding attack on the servers (similar to step 3). Legitimate traffic is started at the beginning of an experiment, and an attack is launched shortly after for a duration of 100 seconds.

Step 6: End

```
mininet# ctrl z (end mininet)
Sudo mn -c (clear topology)
```

IV. Experimental Set-up

In this section, a series of experiments are performed to verify the effects of DoS attack in the SDN network. The experimental setup is shown in Fig. 3. There are two servers and four switches in the network. Each switch has a host connected to it. The Transmission Control Protocol (TCP) server is connected to OpenFlow switch1 while User Datagram Protocol (UDP) server is connected to OpenFlow switch3.

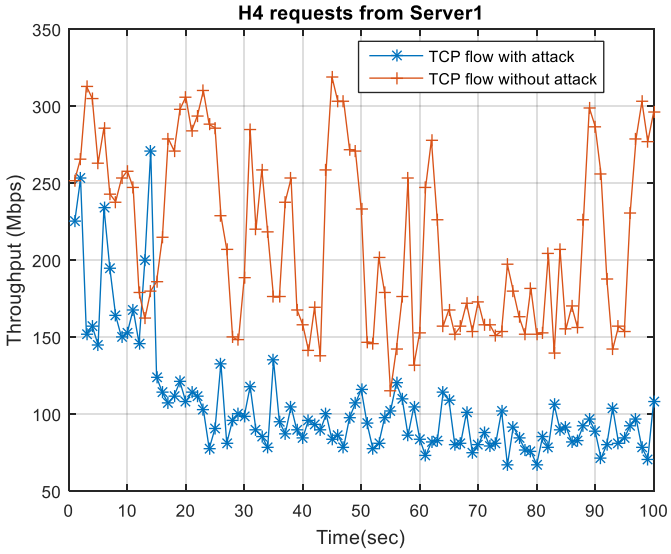

Fig. 4. TCP requests from host 4 to server 1 under ICMP attack.

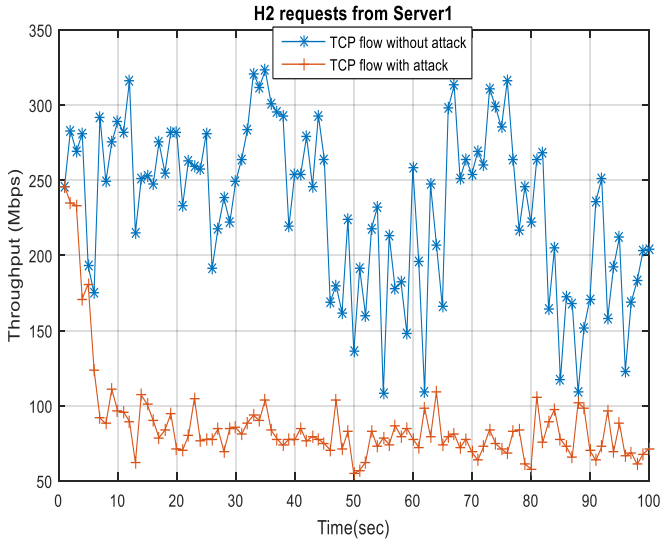

Fig. 5. TCP requests from host 2 to server 1 under ICMP attack.

ICMP flood attack will be launched against both servers by malicious hosts 5 and 6.

V. Results and Analysis

As discussed in the experimental setup, we simulate for two different scenarios; TCP and UDP requests under normal operating condition and under attack. The results for these scenarios are discussed below.

A. *Effect of DoS attack on throughput*

Fig. 4 and Fig. 5 shows a significant drop in throughput due to malicious behaviour (ICMP flood attack) being executed by two attacking nodes. The average throughput for requests made from host 4 to the TCP server1 is 214Mbits/Sec for a total of 2.5Gbytes of information transferred in 100 seconds.

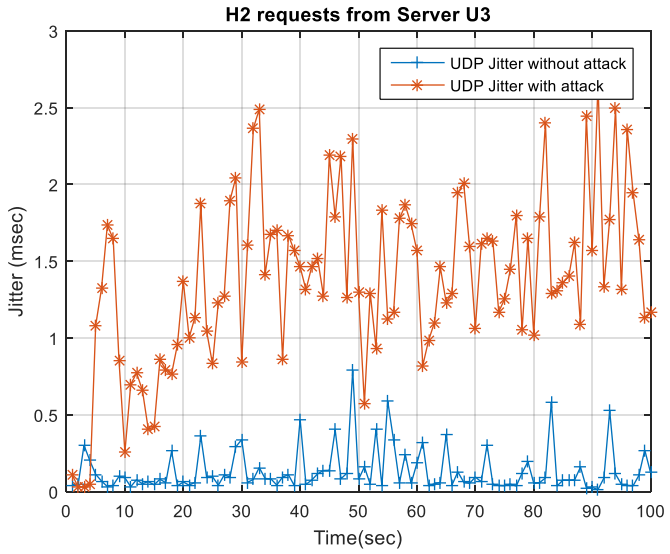

Fig. 6. UDP requests from host 4 to server 1 under ICMP attack.

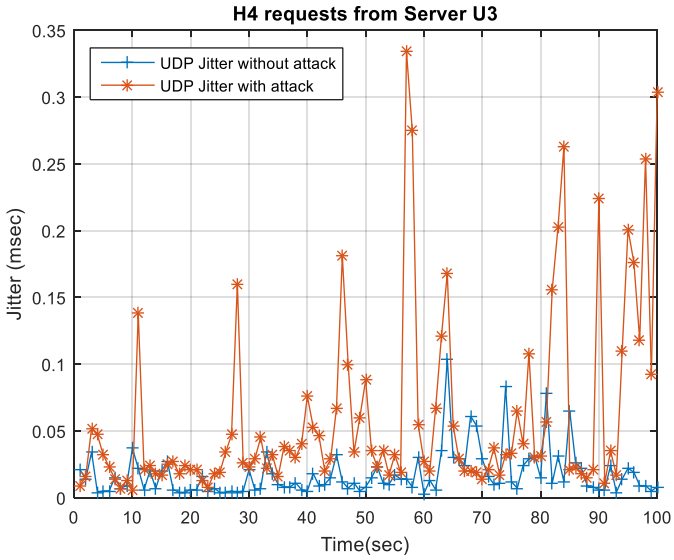

Fig. 7. UDP requests from host 4 to server 1 under ICMP attack.

Similarly, the average throughput of host2 requests from TCP server1 is 233Mbits/sec for a total of 2.72GBytes of information transferred. Notice that Host 2 shows a better bandwidth utilisation than Host 4 and the reason for this is not far-fetched; they are both connected to OpenFlow switch 1. While the better bandwidth utilisation is seen as an advantage here, it is a major security risk and attractive honeypot to launch attack against the server. The impact of this connection is felt when the server is subjected to ICMP flood attack. During attack, the average throughput dropped to 106Mbits/sec from 214Mbits/sec recorded for H4 requests from server1 without ICMP flood attack. The impact of the attack launched by host 5 and 6 became noticeable after 15secs of transmission and the bandwidth utilisation degraded for the rest of the transmission. The trend is similar for host2 requests from TCP server 1 as degradation started after 8secs of transmission and degraded for the rest of the transmission. The average throughput for h2 requests from TCP server1 dropped from 233Mbits/sec to 87.4Mbits/sec when the server is under attack. The degradation is more severe for host 2 when under attack even though higher throughput is recorded during normal operation. Hence, the need for better network design, traffic isolation based on priority for mission-critical network and dynamic proactive ways of addressing DoS attacks when the system is under serious attack.

B. Effect of Dos attack on Jitter

Jitter is defined as a variation in the delay of received packets. In Fig. 6 and Fig. 7, using UDP buffer size of 208 Kbytes, the jitter varies between 0.003ms and 0.789ms. Host 4 Jitter remains within a fair range because it is connected to OpenFlow switch 3 with the UDP server. The spiky delay waveform indicates the presence of congestion in the network. Even though the congestion occurs for a very short period, if the congestion time is more than the scheduled packet transmission time, it will lead to packet drops. Notice that jitter values obtained from host 4 requests to UDP server is better compared to requests from host 2.

VI. Conclusion and Future Work

In this paper, the impact of DoS attack on SDN has been demonstrated. This study reveals that for a simple network, a DoS attack on the infrastructure plane (UDP and TCP servers) will highly degrade network performance as shown in the performance metrics (throughput and jitter). For a Distributed DoS (DDoS) attack with more active agents, the attack will be more severe. Hence, the need for a robust resilient SDN security architecture. While the evaluation of the impact of DoS attack on SDNs remains a very rigorous endeavour, the work carried out in this paper offers a primer to the objective evaluation of DoS attack on SDNs. The simulation results revealed a drop in network throughput from 233Mbps to 87.4Mbps and the introduction of large jitter between 0.003ms and 0.789ms during DoS attack. In the future, the mitigation of DoS and DDoS attacks in an exhaustive way at both control and data plane layers will be examined.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 55–60.

[3] S. Das, G. Parulkar, and N. McKeown, "Rethinking ip core networks," *Journal of Optical Communications and Networking*, vol. 5, no. 12, pp. 1431–1442, 2013.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36–44, 2015.

[6] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[7] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[8] vmware. (2017) Software-defined data center (sddc),. [Online]. Available: http://www.vmware.com/products/nsx/

[9] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[10] P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2016.

[11] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[12] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Computers & security*, vol. 53, pp. 79–108, 2015.

[13] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE transactions on reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[14] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 165–166.

[15] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using protogeni as a launchpad," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–6.

[16] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 413–424.

[17] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 933–939.

[18] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 408–415.

[19] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[20] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 151–152.

[21] TeamMininet. (2017) Mininet. [Online]. Available: http://www.mininet.org/download/

[22] Linux-Foundation-Collaborative-Projects. (2017) Odl. [Online]. Available: https://www.opendaylight.org